WEEKS, SCOTT W., M.S. PK-Hybrid Protocols: A New Framework for Hybrid Encryption. (2011)
Directed by Dr. Stephen R. Tate, 29pp.

This thesis considers cryptographic systems that use public key encryption (PKE) as a building block in a larger system. We call these cryptosystems "PK-hybrid protocols," and develop a framework for describing such protocols that provides a clear way of describing the role of PKE in the PK-hybrid protocol. By clarifying and generalizing the role of PKE in such protocols, we are able to state and prove a powerful lemma for proving the security of PK-hybrid protocols. We then show how this lemma can be used in a proof of security for the standard technique of hybrid encryption, substantially simplifying earlier proofs by Cramer and Shoup [1] and by Abe, *et al* [2].

In addressing these protocols we make improvements to a previously published protocol for Generalized Non-Interactive Oblivious Transfer (GNIOT) due to Gunupudi and Tate [3], and use our new security framework to correct a subtle error in the original security proof provided for GNIOT.

PK-HYBRID PROTOCOLS: A NEW FRAMEWORK FOR HYBRID ENCRYPTION

by

Scott W. Weeks

A Thesis Submitted to
the Faculty of The Graduate School at
The University of North Carolina at Greensboro
in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Greensboro
2011

Approved by

Committee Chair

©2011 Scott W. Weeks

APPROVAL PAGE

This thesis has been approved by the following committee of the Faculty of The Graduate School at The University of North Carolina at Greensboro.

Committee Chair_____

Committee Members_____

_____

_____
Date of Acceptance by Committee

_____
Date of Final Oral Examination

TABLE OF CONTENTS

Page

CHAPTER

I. INTRODUCTION 1

1.1. Basic Terms 2
1.2. Encryption 3

II. DEFINITIONS AND PRELIMINARIES 8

2.1. Formal Security 8
2.2. Other Notions of Security 12

III. PK-HYBRID PROTOCOLS 15

3.1. Plaintext Randomizations and PK-Hybrids 15
3.2. Standard Hybrid Encryption 18

IV. GNIOT 20

4.1. Basic Definitions 20
4.2. Security Game for GNIOT 22
4.3. Improved Solution 24
4.4. Security Proof 25

V. CONCLUSION 27

BIBLIOGRAPHY 28

CHAPTER I

INTRODUCTION

Traditionally, encryption was done through the use of a shared secret, using one of many types of symmetric key encryption (SKE), in which encryption and decryption use the same, shared key. With the advent of long distance communication between computers, it became necessary to communicate securely between two parties that did not necessarily have a previously agreed upon secret. A solution to this problem was devised by Whitfield Diffie and Martin Hellman who introduced the notion of public key encryption (PKE), where two keys exist, one for encryption and a second key for decryption. This provided a strong level of security, but implementations by Diffie and Hellman [4] and Rivest, Shamir and Adleman [5] and others were extremely inefficient, to the point where the slower encryption and decryption speeds made public key encryption schemes impractical for use with large quantities of data. To illustrate, benchmarks of the most popular public key and symmetric key encryption schemes (RSA and AES, respectively) using OpenSSL version 1.0 on an Intel Core 2 6700 processor put the PKE scheme RSA throughput at 778.1 kb/s and SKE scheme AES throughput at 1136 Mb/s. So in this implementation the public key scheme operates at less than one thousandth of the speed of the symmetric encryption scheme.

Hybrid encryption, as described in 1985 by Varadharajan and Sanders [6], provides a solution where public key encryption is used to establish a shared secret, which can then be used as the key for a much faster, more efficient symmetric key

encryption scheme. Hybrid encryption was in common use for a long time, but resisted formal analysis until 2003 when Cramer and Shoup successfully proved that a hybrid scheme using two encryption schemes that were secure against chosen ciphertext attacks (algorithms that are secure even if the adversary is able to request decryption of chosen ciphertext values) [1]. In 2005, Abe, *et al* [2] proved that by adding extra information in the form of a *tag*, security against chosen ciphertext atacks (CCA security), considered to be the strongest form of security for an encryption algorithm, is not necessary for the symmetric encryption scheme and a more relaxed requirement is sufficient.

In this thesis, we generalize hybrid encryption to the concept of the PK-hybrid scheme, which is an encryption scheme that uses public key encryption as a component in a larger, more complex scheme such as the hybrid secret sharing scheme used in Generalized Non-Interactive Oblivious Transfer (GNIOT), described later in this thesis. We then prove that we can limit the possible advantage of an attacker by comparing the PK-Hybrid scheme with a slightly modified version of itself against which the possible advantage is simpler to calculate.

1.1 Basic Terms

Cryptographic systems, or *cryptosystems*, are defined in terms of a suite of functions provided by the system which depend on a security parameter λ that determines protocol characteristics such as key size. *Keysize* is the size of the key, given in terms of the length of the key's binary encoding. *Keyspace* is the set of possible key values. The size of the keyspace is the number of possible key values. For example, if the keysize is 256 bits and all binary strings of length 256 are valid keys, the size of the keyspace is 2^{256}, as that is the total number of keys that can be represented

within that keyspace. *Plaintext* is data that has not been encrypted, such as ASCII characters or HTML form data, although it can also be unstructured binary data. *Ciphertext* is plaintext that has been modifed through some encryption scheme. Note that throughout this thesis, we are assuming that plaintexts and keyspaces are efficiently samplable.

1.2 Encryption

Symmetric Key Encryption

Symmetric key encryption (SKE) is an encryption scheme that has 3 functions: $KeyGen(1^\lambda)$ [1], $Encrypt(k, p)$, and $Decrypt(k, c)$. $KeyGen$ returns a key k with the size being determined by λ. In the theoretical model, keysize can be arbitrarily long, but in practice many SKE implementations have restrictions on what are valid values for λ — for example, the Advanced Encryption Standard (AES) requires keysizes of 128, 192, or 256 bits. In this thesis, as is common practice in cryptography, we assume all algorithms support arbitrarily long keys, so we consider something like AES only in an extension that supports longer keys. $Encrypt : K \times P \to C$ takes a key k from keyspace K and a plaintext value $p \in P$ and returns ciphertext $c \in C$. Likewise, $Decrypt : K \times C \to P$ takes the key k and a ciphertext c and returns a plaintext p. *Encrypt* and *Decrypt* are inverses of each other, that is, $Decrypt(k, Encrypt(k, p))$ will return p. In addition to AES, other examples of SKE systems include the Data Encryption Standard (DES) and most pre-computing ciphers such as the Vigenère Cipher. We can refer to an SKE cryptosystem by a triple that represents these three algorithms, such as $S = (KeyGen, Encrypt, Decrypt)$.

[1] As is standard practice in cryptography, the security parameter is given as a unary representation

Public Key Encryption

Public Key Encryption (PKE), also known as Asymmetric Encryption is an encryption scheme is also defined by three functions: $KeyGen(1^\lambda) : 1^* \to PU \times PR$ where PU is the public keyspace and PR is the private keyspace, $Encrypt : PU \times P \to C$, and $Decrypt : PR \times C \to P$. However, unlike in symmetric encryption there are two keys. One key (the "public key") is used for encryption while the other key (the "private key") is used for decryption. Once again, we can represent a PKE cryptosystem as a triple, $S = (KeyGen, Encrypt, Decrypt)$.

Hybrid Encryption

Hybrid encryption refers to the practice of using a PKE scheme to create a shared secret between two parties that can then be used as the key to an SKE scheme, usually referred to in this context as the *Data Encryption Mechanism (DEM)*. It came into practice because while PKE could be used where two parties did not have a shared secret, it is much slower than SKE and impractical for the amount of data needed for modern online communications. Despite being in widespread use, hybrid encryption resisted a formal security proof for a long period until Cramer and Shoup [1] first published a proof in 2003.

Secret Sharing

Secret sharing is the idea of taking a secret and breaking it into pieces, or *shares*, which can then be used to reconstruct the secret. Brickells [7] defined secret sharing as follows. Consider a secret s. The dealer who holds this secret creates a set of shares of s and distributes them to a set of n participants. Let Γ be a set of subsets of the participants. A *secret sharing scheme* for Γ is a method of distributing shares

to each member of n such that any subset of participants that is an element of Γ can reconstruct the secret, while any subset of participants that is not an element of Γ cannot. If any subset of participants that is not an element of Γ cannot determine any information about the secret, then the secret sharing scheme is said to be perfect. Secret sharing schemes were first constructed by Blakley [8] and Shamir [9]. They created what are called *threshold schemes* because given n shares, there is some k such that any set of k shares is sufficent to reconstruct the secret. The simplest example of this is as follows:

Consider a key that we want to split into two shares, both of which will be needed to recreate the key. We can randomly generate the first share, and then to obtain the second share we use an XOR operation to combine the first share and the key, which produces the second share. Both shares, XORed together will create the original key, but possessing only one share provides no information about the key because all valid values for the key are equally valid. As an example consider the key 1001001001 and the first share which we will choose to be 0100100111. The XOR operation provides us with the second share: 1101101110. Using XOR to combine the two shares produces the original key, but note that either share by itself can be used as the share for any other key by XORing the share with the new key. Thus, since no information can be obtained from a single share, we say that this is a secret sharing scheme with perfect security.

PK-Hybrid

We define a PK-hybrid scheme as any cryptosystem where public key encryption is used as a component in a larger, more complex cryptosystem. Standard hybrid encryption, as described above, is an example of this, as is a scheme where PKE is used to encrypt shares from a secret sharing scheme, as in the GNIOT scheme

described later. As an example, consider an online game whose victory condition involves reconstructing a secret that has been broken into shares. Each share is encrypted with a public key scheme, and as the player satisfies game conditions, he may make decryption requests from the central server, which sends plaintext shares to the user. In this example, the security of the game system as a whole requires that the PKE-encrypted shares provide no advantage to the player.

Contributions of this thesis

The work in this thesis provides several contributions to the cryptographic research literature, which we list below:

- We define the concept of a PK-hybrid scheme, which is a different, broader way of looking at hybrid encryption and similar cryptosystems.
- We present a powerful lemma that can be used to prove the security of a PK-hybrid system more easily than has been done in the past.
- We make improvements to a previously published protocol for GNIOT due to Gunupudi and Tate, and use our new security framework to correct a subtle error in the original security proof provided for GNIOT.

Outline

Chapter II introduces terms and concepts necessary to understand the main body of the thesis. Chapter III introduces the concept of a plaintext randomization of a PKE security scheme and states the main lemma of this thesis. The lemma is then used to prove the security of standard hybrid encryption. Chapter IV reproduces the concept of Generalized Non-Interactive Oblivious Transfer from Gunupudi and

Tate [3] and provides an improved solution and proof using the new framework we introduced in Chapter III. Chapter V summarizes the contributions of this thesis and describes further work to be done.

CHAPTER II

DEFINITIONS AND PRELIMINARIES

In this chapter we present formal definitions related to the security of cryptosystems that we discuss in this chapter. For more extensive coverage, the reader is referred to a recent text on cryptography such as the series by Goldreich [10] or the textbook by Katz and Lindell [11].

2.1 Formal Security

This section provides basic terminology and concepts from modern cryptography, focused on basic encryption techniques.

Terms

When describing cryptosystems, security is often defined in terms of a game between two parties which are referred to as the adversary and the oracle. An *Oracle* is an algorithm that runs the game and has information or capabilities that the adversary does not have access to. It is important to note that the oracle is a separate party and the adversary does not duplicate the functionality. A standard type of oracle when describing a cryptosystem is an encryption oracle which would have one function, $\mathcal{O}.Encrypt(\cdots)$ where the function would take some number of variables including a plaintext to be encrypted. Likewise, a decryption oracle would have the function $\mathcal{O}.Decrypt(\cdots)$ and would take a ciphertext input. When talking about the security of a cryptosystem against an attacker, we call the attacker the *Adversary*.

Games

Security of cryptosystems is often defined in terms of a game between a probabilistic polynomial time (PPT) adversary and a game oracle. The oracle sets internal, persistent state variables, and answers queries for the adversary. The game is defined in terms of the interface to the game oracle, as a set of functions of the following form.

- $G.Initialize(\lambda)$: Sets up persistent variables that are maintained throughout the game. Typically this can involve generating a random key, and picking a random bit that the adversary will be tasked with guessing.
- $G.OracleQuery()$: One or more functions are defined that allow the adversary to query the oracle.
- $G.IsWinner(a)$: Takes a value a from the adversary at the end of the game, and returns true or false depending on whether the adversary's answer a is a winning answer.

As an example, one version of security for symmetric encryption systems is referred to as indistinguishability under chosen plaintext attacks (written "IND-CPA security"), which for an SKE system $S = (\mathsf{KenGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is defined with the following game.

- $\mathsf{SKE\text{-}CPA}^S.\mathsf{Initialize}(\lambda)$: The oracle generates an SKE key $k = S.KeyGen(1^\lambda)$ and picks a random bit $b \in \{0, 1\}$.
- $\mathsf{SKE\text{-}CPA}^S.\mathsf{Encrypt}(x)$: The oracle sets $c = S.Encrypt(k, x)$ and returns c to the adversary.

- $\mathsf{SKE\text{-}CPA}^S.\mathsf{Challenge}(x_0, x_1)$: The oracle sets $c = Encrypt(k, x_b)$ and returns c to the adversary. *Challenge* may only be called once — any further calls return null.

- $\mathsf{SKE\text{-}CPA}^S.\mathsf{IsWinner}(g)$: Takes a bit g from the adversary, and returns true if and only if $g = b$.

The adversary is given access to $S.KeyGen$ and $S.Encrypt$ and is allowed to call each of them polynomially many times at any point during the game. The game itself progresses as follows:

1. $\mathcal{O}$ calls $S.KeyGen(\lambda)$ and randomly chooses bit $b \in \{0, 1\}$.

2. The adversary may make polynomially many calls to *Encrypt* and may make any calculations it wishes.

3. A chooses 2 plaintext values (x_0, x_1) and sends them to the oracle $\mathcal{O}$. These are the *challenge plaintexts*.

4. $\mathcal{O}$ calls $S.Encrypt(x_b, k)$.

5. The encrypted value c is returned to A. This is the *ciphertext*.

6. A can make additional encryption queries and any calculations it wishes and then outputs guess g.

7. A wins if $g = b$.

Since it is trivial to design an adversary that can win the SKE game 50% of the time by simply guessing randomly, we talk about the adversary's *advantage*

against the game. The advantage of A against game G, written $Adv_{A,G}$ is the following:

$$Adv_{A,G} = |P(A \text{ wins }) - \frac{1}{2}|$$

That is to say, the difference of the probability of the adversary winning and $\frac{1}{2}$, the probability of winning by random chance. It is an absolute value because an adversary that consistently loses can be trivially converted into one that wins by using it as normal and simply inverting the guess. The advantage against arbitrary probabilistic polynomial adversaries is bounded by Adv_G, which is defined as

$$Adv_G = \sup_A(Adv_{A,G})$$

where the supremum is taken over all probabilistic polynomial time adversaries.

Perfect Security

Perfect Security is the term for a cryptosystem where the best possible advantage for the adversary is zero. That is, where it is not possible to obtain better results than random chance. In this thesis we discuss secret sharing, where perfect security is possible to achieve. The simplest example of this is the XOR based 2-of-2 threshold secret sharing scheme discussed above which has perfect security. Note that in the example an adversary trying to break the scheme has no information about the unknown share — until meeting the threshold every possible value is still equally valid.

Negligable Security

In cryptography it is usually not possible to achieve perfect security — many games can be won if the adversary knows a secret key known to the oracle, and in these

cases it may be possible for the adversary to simply guess the key value to win the game. Consider an adversary A that plays the IND-CPA SKE game presented above against a cryptosystem with keyspace $K = \{0,1\}^\lambda$. A plays by randomly generating a key value, trying a decryption, and then either outputting the correct answer (if the decryption succeeds) or randomly guessing (if the decryption fails). The probability that A guesses the key correctly is $2^{-\lambda}$, so in this example the advantage $Adv_{A,SKE} = 2^{-\lambda}$. Thus, when discussing most cryptosystems instead of perfect security our goal is a system in which the advantage is *negligable*. Since in the theoretical model all keysizes can increase indefinitely, we consider the security of an algorithm as a function of keysize. The formal definition of a negligible function is a function $f(\lambda) : \mathbf{Z} \to \mathbf{R}$ such that for every positive integer c_0 there exists integer c_1 such that for all $\lambda > c_1$,

$$|f(\lambda)| < \frac{1}{\lambda^{c_0}}$$

For example, $\frac{1}{2^\lambda}$ is negligable, but $\frac{1}{\lambda^2}$ is not.

2.2 Other Notions of Security

When discussing the security of a cryptosystem, there are standardised notionsls of security that have been developed. The two that are relevant to us are CPA-security (which was defined previously) and CCA-security. The definitions are presented with respect to symmetric encryption, but the conversion to public key schemes is trivial.

IND-CCA Security for Symmetric Encryption

Chosen ciphertext security is defined similarly to chosen plaintext security, but with the addition of a decryption oracle. For this to be sensible, the decryption oracle

cannot allow the adversary to request decryption of the challenge ciphertext, so this is reflected in the definition.

- $\mathsf{SKE\text{-}CCA}^S.\mathsf{Initialize}(\lambda)$: The oracle generates an SKE key $k = S.\mathsf{KeyGen}(1^\lambda)$, picks a random bit $b \in \{0,1\}$, and sets cc (for "challenge ciphertext") to null.
- $\mathsf{SKE\text{-}CCA}^S.\mathsf{Encrypt}(x)$: The oracle sets $c = S.\mathsf{Encrypt}(\mathsf{k},\mathsf{x})$ and returns c to the adversary.
- $\mathsf{SKE\text{-}CCA}^S.\mathsf{Decrypt}(x)$: If $x = cc$, return null to the adversary. Otherwise, the oracle sets $p = S.\mathsf{Decrypt}(\mathsf{k},\mathsf{x})$ and returns p to the adversary.
- $\mathsf{SKE\text{-}CCA}^S.\mathsf{Challenge}(x_0, x_1)$: The oracle sets persistent state variable $cc = S.\mathsf{Encrypt}(\mathsf{k},\mathsf{x_b})$ and returns cc to the adversary. Challenge may only be called once — any further calls return null.
- $\mathsf{SKE\text{-}CCA}^S.\mathsf{IsWinner}(g)$: Takes a bit g from the adversary, and returns true if and only if $g = b$.

The Left-Right PKE Game

The left-right PKE game (LR-PKE), presented in [12] is similar to standard CCA games with a few modifications to allow for multiple, consistent challenge encryptions. The name comes from the fact that the challenge call consistently encrypts either the left or right argument, and the task of the adversary is to decide which it is.

- $\mathsf{LR\text{-}CCA}^S.\mathsf{Initialize}(\lambda)$: The oracle generates an SKE key $k = S.\mathsf{KeyGen}(1^\lambda)$, picks a random bit $b \in \{0,1\}$ and sets C as a set of null challenge ciphertexts that is allowed to grow polynomially large.

- $\mathsf{LR\text{-}CCA}^S.\mathsf{Encrypt}(x)$: The oracle sets $c = S.\mathsf{Encrypt}(\mathsf{k}, \mathsf{x})$ and returns c to the adversary.

- $\mathsf{LR\text{-}CCA}^S.\mathsf{Decrypt}(x)$: If $x \in C$, return null to the adversary. Otherwise, the oracle sets $p = S.\mathsf{Decrypt}(\mathsf{k}, \mathsf{x})$ and returns p to the adversary.

- $\mathsf{LR\text{-}CCA}^S.\mathsf{Challenge}(x_0, x_1)$: The oracle calculates $c = S.\mathsf{Encrypt}(\mathsf{k}, \mathsf{x_b})$, adds c to C, and returns c to the adversary. Challenge may be called polynomially many times.

- $\mathsf{LR\text{-}CCA}^S.\mathsf{IsWinner}(g)$: Takes a bit g from the adversary, and returns true if and only if $g = b$.

CHAPTER III

PK-HYBRID PROTOCOLS

In this chapter we introduce the key concept behind our improved approach to PK-hybrid protocol analysis, the notion of *plaintext randomization*. This allows us to state and prove a powerful lemma for proving security of PK-hybrid cryptosystems.

3.1 Plaintext Randomizations and PK-Hybrids

Consider an adversary A that plays a generic game G against oracle $\mathcal{O}$ that uses CCA-secure PKE encryption scheme S_0. Consider the following modifications to S_0 which we will refer to as the plaintext randomization of S_0, and denote S_1:

- When $S_1.Encrypt(pk, x)$ is called, S_1 creates $c = PKE_{pk}(y)$ where y is a randomly generated, valid plaintext of the same length as x. S_1 stores the tuple (c, x) and returns c.
- When $S_1.Decrypt(sk, c)$ is called, S_1 returns x if (c, x) is stored and decrypts normally otherwise.

Let us refer to the game using S_0 as G^{S_0} and the game using S_1 as G^{S_1}.

Lemma 3.1.1 *Let G be a game that uses a public key encryption scheme S_0 and let S_1 be the plaintext randomization of S_0. Then, for any probabilistic polnomial time adversary A, $|Adv_{A,G^{S_0}} - Adv_{A,G^{S_1}}| \leq 2Adv_{LR^{S_0}}$*

Proof:

Let us construct A' that plays the standard LR-PKE game against oracle $\mathcal{O}_{\mathcal{LR}}$. During intialization, $\mathcal{O}_{\mathcal{LR}}$ chooses bit b to determine right or left. Let A be the adversary from the theorem statement that plays game G against $\mathcal{O}_{\mathcal{G}}$ using PKE encryption scheme S_0 which is being simulated by A'. Whenever there is a call $S_0.Encrypt(pk, x)$, A' does the following:

- A' generates random plaintext value y
- A' submits (x, y) as a plaintext pair to $\mathcal{O}_{LR^{S_0}}$ for encryption.
- A' receives ciphertext c from $\mathcal{O}_{LR^{S_0}}$ and stores (x, c)
- A' continues its simulation of $\mathcal{O}_G$ using c as the output from $S_0.Encrypt$

When there is a valid call to decryption function $S_0.Decrypt(pk, y)$, A' returns x if (x, y) is stored, and performs the decryption normally otherwise. When A outputs its final value such as a guess g, A' determines whether or not A wins in its simulated game G. Finally,

- If A wins, A' outputs "left" (guessing that $b = 0$)
- If A loses, A' outputs "right" (guessing that $b = 1$)

Thus, A' wins if A wins and $b = 0$, or if A loses and $b = 1$.

If $b = 0$ then A was playing G^{S_0} and if $b = 1$ then A was playing G^{S_1}. Therefore, the probability that A' wins is as follows:

$$\begin{aligned} P(A' \text{ wins }) &= \frac{1}{2}P(A \text{ wins } | b = 0) + \frac{1}{2}P(A \text{ loses } | b = 1) \\ &= \frac{1}{2}(P(A \text{ wins } | b = 0) + \frac{1}{2}(1 - P(A \text{ wins } | b = 1)) \\ &= \frac{1}{2} + \frac{1}{2}P(A \text{ wins } | b = 0) - \frac{1}{2}P(A \text{ wins } | b = 1) \end{aligned}$$

Since $Adv_{A',LR} = |P(A' \text{ wins } - \frac{1}{2})|$, we know that

$$Adv_{LR} \geq |\frac{1}{2}P(A \text{ wins } | b = 0) - \frac{1}{2}P(A \text{ wins } | b = 1)|$$

Thus

$$|P(A \text{ wins } | b = 0) - P(A \text{ wins } | b = 1)| \leq 2Adv_{LR}$$

If $b = 0$, A is playing G^{S_0} and therefore

$$P(A \text{ wins } | b = 0) = P(A \text{ wins against } G^{S_0})$$

Likewise, if $b = 1$, A is playing G^{S_1} and therefore

$$P(A \text{ wins } | b = 0) = P(A \text{ wins against } G^{S_1})$$

Since for all a and b, $||a| - |b|| \leq |a - b|$, we can bound $|Adv_{A,G^{S_0}} - Adv_{A,G^{S_1}}| = ||P(A \text{ wins } | b = 0) - \frac{1}{2}| - |P(A \text{ wins } | b = 1) - \frac{1}{2}|| \leq |P(A \text{ wins } | b = 0) - P(A \text{ wins } | b = 1)|$ and by the above bound we can say:

$$|Adv_{A,G^{S_0}} - Adv_{A,G^{S_1}}| \leq 2Adv_{LR}$$

∎

3.2 Standard Hybrid Encryption

To demonstrate the use of this lemma, we use it to prove standard hybrid encryption:

Theorem 3.2.1 *If T is a CCA-secure symmetric encryption scheme and S_0 is a CCA-secure public-key encryption scheme with CCA advantages given by Adv_{CCA^T} and $Adv_{CCA^{S_0}}$ respectively, then the hybrid encryption system $H_{S_0,T}$ that combines the PKE and SKE encryption schemes is CCA-secure with advantage bounded by*

$$Adv_{CCA^H} \leq 2Adv_{LR^{S_0}} + Adv_{CCA^T}$$

Proof:

Let $H_{S_0,T}(x)$ represent the plaintext x protected by H using PKE scheme S_0 and SKE scheme T, and let S_1 be the plaintext randomization of S_0. $H_{S_1,T}(x)$ therefore consists of a random, useless key and the SKE encryption of the plaintext x with a random, independent key. Since the PKE data contains no information about the SKE encryption (unlike in the standard game where it would contain the encrypted SKE key), we can say that breaking $H_{S_1,T}(x)$ is equivalent to breaking the SKE scheme. Thus the advantage against $H_{S_1,T}(x)$ is

$$Adv_{H_{S_1,T}(x)} \leq Adv_T$$

By lemma 3.1.1 we know that

$$|Adv_{A,H_{S_0,T}} - Adv_{A,H_{S_1,T}}| \leq 2Adv_{LR^{S_0}}$$

as before we can limit bound this as

$$||Adv_{A,H_{S_0,T}}| - |Adv_{A,H_{S_1,T}}|| \leq 2Adv_{LR^{S_0}}$$

and therefore

$$|Adv_{A,H_{S_0,T}}| \leq 2Adv_{LR^{S_0}} + Adv_T$$

■

CHAPTER IV

GNIOT

Gunupudi and Tate [3] defined the Generalized Oblivious Transfer (GOT) and Generalized Non-Interactive Oblivious Transfer (GNIOT) problems which are useful in many secure function evaluation applications, such as supporting secure mobile agents. In this thesis we use the same basic problem definitions but provide an improved solution and security proof.

4.1 Basic Definitions

The basic definitions from Gunupudi and Tate are reproduced below.

Definition 4.1.1 (GOT) *Define λ as the security parameter and l_d as the length of the data items being sent by Alice to Bob. Assume that Alice has n data sets $S_1, S_2, \cdots, S_n$, with values $x_{i,j} \in \{0,1\}^{l_d}$ for $i \in \{1, 2, \cdots, n\}$ and $j \in \{1, 2, \cdots, m_i\}$, and parameters $k_1, k_2, \ldots, k_n$, where $1 \leq k_i \leq m_i$. At the end of the GOT execution, Bob will have either no result (represented by $\perp$) or a set of exactly k_i values of his choice from each set S_i, for $i \in \{1, 2, \cdots, n\}$.*

We will need to refer to sets of indices into the data set, so define index set I to be a set of indices (i, j), and define $I(i) = \{j \mid (i, j) \in I\}$. With respect to the parameters provided in an instance of GOT, we say that index set I is *well-formed* if $|I(i)| = k_i$ for all $i \in \{1, \ldots, n\}$.

We define GNIOT as a set of operations which perform GOT, but accomplish this task without requiring any interaction between the receiver and any other party

after the receiver decides which values he wants. For maximum flexibility, allowing either batched or individual decryptions, we define the decryption operation as a stateful process which is called repeatedly — only at the very end are we required to have the actual plaintext values.

Definition 4.1.2 (GNIOT) *Generalized Non-Interactive Oblivious Transfer consists of the following phases, which provide a solution to the GOT problem.*

Setup Phase. *This phase involves key generation. Given security parameter λ, the key generation algorithm returns*

$$(K_p, K_s) \leftarrow Setup(1^\lambda)$$

where K_p is the public key information, and K_s is the secret key information.

Transmit Phase. *This phase transforms the set of values $x_{i,j} \in \{0,1\}^{l_d}$ for $i \in \{1, 2, \cdots, n\}$ and $j \in \{1, 2, \cdots, m_i\}$ into a data blob which can be transmitted to the receiver. Specifically,*

$$C \leftarrow Transmit_{K_p} \begin{pmatrix} \langle k_1, x_{1,1}, x_{1,2}, \cdots, x_{1,m_1} \rangle, \\ \langle k_2, x_{2,1}, x_{2,2}, \cdots, x_{2,m_2} \rangle, \\ \vdots \\ \langle k_n, x_{n,1}, x_{n,2}, \cdots, x_{n,m_n} \rangle \end{pmatrix}.$$

Decrypt Phase. *In this phase, the receiver gives the indices (i, j) of the $x_{i,j}$ values that he wishes to receive. The state-based process begins by calculating the initial state $S_0 \leftarrow InitialState(C)$, and then evolving the state and providing answers to queries as*

$$(t_k, S_k) \leftarrow Decrypt_{K_s}(S_{k-1}, C, i_k, j_k),$$

for $k = 1, 2, \ldots, q$ for some number of queries q. We require that index information be embedded in t_k such that there is a function "ind" that extracts this information as

$$(i_k, j_k) \leftarrow ind(t_k).$$

PostProcess Phase. *This phase takes the results of the Decrypt calls and either fails (giving $\perp$ as the result) or produces q plaintext values as*

$$\langle v_1, v_2, \ldots, v_q \rangle \leftarrow PostProcess(t_1, t_2, \cdots, t_q)$$

4.2 Security Game for GNIOT

In the GNIOT game, adversary A sends a set of plaintext challenge texts arranged as a set of n rows of m_i pairs each. The oracle $\mathcal{O}$ independently chooses one plaintext from each pair and calls $Transmit$ with the chosen values. The adversary is allowed to make k_i decrypt queries for each row i, and at the end attempts to guess the bit corresponding to any challenge pair that *Decrypt* was not requested on.

- $GNIOT.Initialize(\lambda)$: Call *Setup* and get (K_p, K_s) and randomly choose bits $r \in \{0, 1\}$ and $b_{i,j} \in \{0, 1\}$, $1 \leq i \leq n$ and $1 \leq j \leq m_i$. Finally, for each i set k_i.

- $GNIOT.Challenge(X)$: X is a set of plaintext challenge pairs in the following format:

$$X = \begin{pmatrix} (x^0_{1,1}, x^1_{1,1}) & \cdots & (x^0_{1,m}, x^1_{1,m_1}) \\ \cdots & \cdots & \cdots \\ (x^0_{n,1}, x^1_{n,1}) & \cdots & (x^0_{n,m}, x^1_{n,m_n}) \end{pmatrix}$$

Using the input it constructs

$$X' = \begin{pmatrix} k_i & x_{1,1}^{b_{1,1}} & \cdots & x_{1,m}^{b_{1,m}} \\ \cdots & \cdots & \cdots & \cdots \\ k_i & x_{n,1}^{b_{n,1}} & \cdots & x_{n,m}^{b_{n,m}} \end{pmatrix}$$

It then calls $Transmit_{K_p}(X')$.

- $GNIOT.Decrypt(i, j)$: Calls $Decrypt(i, j)$
- $GNIOT.IsWinner(i, j, g)$: Returns **true** if $GNIOT.Decrypt(i, j)$ has not been called and $g = b_{i,j}$. Returns **false** otherwise.

1. Oracle $\mathcal{O}$ initializes the game by calling $S.KeyGen$ and randomly choosing $r \in \{0, 1\}$ and $b_{i,j} \in \{0, 1\}$, $1 \leq i \leq n$ and $1 \leq j \leq m_i$.
2. A chooses plaintext pairs

$$\begin{pmatrix} (x_{1,1}^0, x_{1,1}^1) & \cdots & (x_{1,m}^0, x_{1,m_1}^1) \\ \cdots & \cdots & \cdots \\ (x_{n,1}^0, x_{n,1}^1) & \cdots & (x_{n,m}^0, x_{n,m_n}^1) \end{pmatrix}$$

3. $\mathcal{O}$ generates and returns C where

$$C = \begin{pmatrix} (c_{1,1}, c_{1,1}) & \cdots & (c_{1,m}, c_{1,m_1}) \\ \cdots & \cdots & \cdots \\ (c_{n,1}, c_{n,1}) & \cdots & (c_{n,m}, c_{n,m_n}) \end{pmatrix}$$

and $c_{i,j} = S.Encrypt(pk, x_{i,j}^{b_{i,j}})$.

4. A is allowed to decrypt k_i values on each line.
5. A is free to perform any computations using the information it obtained.
6. A outputs guess g and an index i, j.
7. A wins if $g = r$.

4.3 Improved Solution

In this section we provide an improved solution to the GNIOT problem that uses our PK-hybrid framework to enable a simpler proof than in the original. S refers to the PKE scheme in use and T refers to the SKE scheme. In addition, the original solution embedded encrypted data inside the PKE, limiting the size of the data.

- $GNIOT.Setup(\lambda)$: Call $S.KeyGen(\lambda)$ receiving (pk, sk). Return (pk, sk).
- $GNIOT.Transmit(X)$:
    1. Choose key R, which is broken into n of n shares $R_1, \cdots, R_n$ using a perfect secret sharing scheme.
    2. Create keys $k_{i,j}$ and breaks each into 2 shares using a perfect secret sharing scheme so that the shares of $k_{i,j}$ are R and some $R_{i,j}$.
    3. Each R_i is broken into k_i of m_i shares $s_{i,1}, \cdots, s_{i,m_i}$ using a perfect secret sharing scheme.
    4. Return $C =$

$$\begin{pmatrix} (PKE(s_{1,1}, R_{1,1}), SKE_{k1,1}(x_{1,1})) & \cdots & (PKE(s_{1,m}, R_{1,m_1}), SKE_{k1,m_1}(x_{1,m_1})) \\ \cdots & \cdots & \cdots \\ (PKE(s_{n,1}, R_{n,1}), SKE_{kn,1}(x_{n,1})) & \cdots & (PKE(s_{n,m}, R_{n,m_n}), SKE_{kn,m_n}(x_{n,m_n})) \end{pmatrix}$$

- $GNIOT.Decrypt(i, j)$: Returns $t = (i, j, x_{i,j}^{b_{i,j}})$ if less than k_i decryptions have been called on row i. Returns null otherwise.
- $GNIOT.PostProcess(T)$: Takes a set of $Decrypt$ outputs T of size q and outputs plaintext values $(x_1, x_2, \cdots, x_q)$.

4.4 Security Proof

The security proof provided by Gunupudi and Tate had a subtle error in one case of the proof. Using the framework provided by this thesis allows us to develop a simpler and corrected proof.

Theorem 4.4.1 *Given an adversary A that plays an unmodified version of the GNIOT game,* G^{S_0}, $Adv_{A,G^{S_0}} \leq 2Adv_{LR} + Adv_{SKE}$.

Proof:

Let us consider the GNIOT game as 2 distinct oracles, the main game oracle and the PKE encryption/decryption oracle S_0. Let us also consider a modified PKE oracle S_1 and an adversary A that attacks the GNIOT game.We will represent the two games as G^{S_0} and G^{S_1} respectively.

The attacker is attempting to gain information about data that is protected by the SKE scheme. The key to the SKE scheme is protected by secret sharing, and the shares are protected by the PKE scheme. Since the secret sharing schemes in use are perfect, the attacker gains no advantage from obtaining a set of shares that does not meet the threshold to recreate the key. In game G^{S_1}, the key to the SKE scheme is broken into shares, but those shares are not provided to the attacker even in encrypted form, thus $Adv_{A,G^{S_1}} \leq Adv_{SKE}$. That is to say, the key cannot be recreated even if the PKE scheme were to be broken, thus the attacker's advantage is at best the best possible advantage against the SKE encryption scheme. Since the SKE scheme is CCA-secure, Adv_{SKE} is negligable and therefore so is $Adv_{A,G^{S_1}}$. By Theorem 3.1.1 we know that

$$|Adv_{A,G^{S_0}} - Adv_{A,G^{S_1}}| \leq 2Adv_{LR}$$

As was shown above, this is equivalent to

$$|Adv_{A,G^{s_0}} - Adv_{SKE}| \leq 2Adv_{LR}$$

Therefore

$$Adv_{A,G^{s_0}} \leq 2Adv_{LR} + Adv_{SKE}$$

∎

CHAPTER V

CONCLUSION

In this thesis we defined the concept of a PK-hybrid scheme and used this as a broader, more flexible framework to be applied to cyptosystems that incorporate PKE. We also proved a powerful lemma that, combined with our new framework, can be used a general tool for proving the security of PK-hybrid cryptosystems more easily than has been done in the past, and as an example proved that the standard hybrid encryption technique is CCA-secure if the component cryptosystems (PKE and SKE) are CCA-secure. Finally, we made improvements to the work previously published by Gunupudi and Tate [3] using our new framework to construct a simpler solution that is easier to prove secure.

Future work includes creating a stronger version of the lemma that restricts the advantage against the PK-hybrid cryptosystem by $Adv_{CCA-PKE}$ instead of Adv_{LR-PKE}. Using this, we could unify results that compare advantage of LR to CCA with our PK-hybrid framework. Also, the original GNIOT paper used trusted computing hardware capabilities, and work could be done putting this work into that context. Finally, the secret sharing results can be made to be more general, instead of specific to the GNIOT scheme.

BIBLIOGRAPHY

[1] R. Cramer and V. Shoup, "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," *SIAM J Comput.*, vol. 33, pp. 167–226, 2003.

[2] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup, "Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM," *Cramer, R.J.F. (ed.) EUROCRYPT 2005*, vol. 3494, pp. 128–146, 2005.

[3] V. Gunupudi and S. R. Tate, "Generalized non-interactive oblivious transfer using count-limited objects with applications to secure mobile agents," *Lecture Notes in Computer Science*, vol. 5143/2008, pp. 98–112, 2008.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography," 1976.

[5] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[6] V. Varadharajan and P. W. Sanders, "Practical secure electronic mail system with public key distribution," *Computer Communications*, vol. 8, pp. 121–127, 1985.

[7] E. F. Brickell, "Some ideal secret sharing schemes," *Advances in Cryptology EUROCRYPT 89*, vol. 434/1990, pp. 468–475, 1990.

[8] G. R. Blakley, "Safeguarding cryptographic keys," *Proceedings AFIPS 1979 National Computer Conference*, pp. 313–317, 1979.

[9] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22(11), pp. 612–613, 1979.

[10] O. Goldreich, *Foundations of Cryptography.* Cambridge University Press, 2003-2004, vol. 1-2.

[11] J. Katz and Y. Lindell, *Introduction to Modern Cryptography: Principles and Protocols.* Chapman and Hall/CRC, 2007.

[12] M. Bellare, A. Boldyreva, and S. Micali, "Public-key encryption in a multi-user setting: Security proofs and improvements," *Advances in Cryptology - Eurocrypt 2000 Proceedings, Lecture Notes in Computer Science*, vol. 1807, pp. 259–274, 2000.